IJAM International Journal of Agile Manufacturing

Foundations of an Agile Design Methodology

P. C. Matthews*, C. D. W. Lomas*, N. D. Armoutis* and P. G. Maropoulos[†]
**School of Engineering, University of Durham, Durham, UK*
[†]Dept. of Mechanical Engineering, University of Bath, Bath, UK
E-mail: p.c.matthews@durham.ac.uk

Abstract: Modern design projects typically are undertaken concurrently in a virtual enterprise network of expert design and manufacturing agents. The general need for agile response in turbulent environments is well documented and has been analysed at the manufacturing phase. This paper proposes a framework to enable analysis of an agile design methodology. This models the occurrence of an unexpected event in a concurrent design project. The redistribution of the design work can be controlled within the virtual enterprise, and the total redistribution impact can be measured. A four-level classification scheme for the severity of unexpected events is proposed. Each event level is illustrated with a design scenario. A Monte Carlo-based simulation tool is proposed for work redeployment policy analysis.

Key Words: Agile, Design, Unexpected Events, Concurrent Engineering.

1. Introduction

The concepts of Agility and Design are well defined. Agility is regarded as the ability to react rapidly to changes in the environment, whether expected or not. When applied to manufacturing, this is regarded as the ability to change a manufacturing resource rapidly so as to produce a different assembly or product. Thus, an agile manufacturer is able to respond rapidly to changes in product demand. Design is the process of transforming a set of potentially ill-defined customer requirements into a physical product. This process is a combination of determining appropriate attributes and then determining their values.

A number of methodologies exist for deployment in different application areas and in different corporate cultures. As designs have become more complex, it has been necessary to migrate to concurrent design methods [1]. Concurrent design methods, however, lack the ability to respond to unpredicted changes in environmental conditions.

The need for diverse design teams is a shared property between Agile Design and Concurrent Design. These are both instances of Virtual Enterprises, as they both lever resource networks to solve a design problem [2]. The distinction between these two methodologies is the responsiveness that is core to Agile Design. Given a design brief, both methods use competence profiling to allocate design tasks suitably to a pool of design agents. When an unpredicted event occurs, which could be either a late customer request, the failure of an agent, or some other external environmental impact, the concurrent design process is interrupted. The impact of the event is evaluated, and the design process is restructured accordingly but with minimal impact to unaffected agents. The aim is to minimise the total impact to the design process. At the same time, however, it is important to minimise communication between active agents as this delays progress. This is achieved not only by reviewing the directly impacted agents, but also by their interfacing agents and their associated latencies. These two contradicting requirements provide the basis for optimising the management of the concurrent design process, thus providing the agility.

This paper will develop a set of foundations that will define Agile Design and provide a set of requirements that must be met to enable Agile Design. The following three distinct Agile Design scenarios are classified: (1) Late customer design brief change; (2) In-production design modification; and (3) Design Agent failure to meet requirement. Each scenario requires an agile reaction to ensure timely production. It also should be noted that while these scenarios appear to be "negative" events, there are equivalent positive scenarios (e.g., the identification of new tools that allow for more rapid design). These events, however, still will require a degree of design review, and so also can be analysed using this framework.

2. Background

At its most basic, the design process takes us from an initial requirements specification through to the final set of manufacturing plans [3]. This is represented by a single work thread, marked by a set of check points representing the design moving from one development phase to the next. It is at these check points wherein unsatisfactory work can be referred back, either to the start of that phase or further back if more significant errors are found. These design phases can be seen to consume time as a primary resource. This simple representation, however, does not provide for concurrency of work. As such, it forms the terminal part of a concurrent engineering project.

Concurrent engineering is the distribution of the design work, as well as the potential manufacturing work, among a number of agents [4]. An agent in this case will be a design team, manufacturing shop, assembly facility, or some other related facility. These agents then either can apply concurrent engineering again, recursively, or they can follow the basic linear design process if they are a "terminal" agent. For example, the design team might sub-contract some of the design work to another specialist design group. The agents are selected according to their known expertise [5]. These agents are networked through a virtual enterprise while the project is underway and are combining the work toward the end of the project. Through this network, agents communicate as necessary. This concurrent engineering approach provides a means for rapidly creating enterprises with high degrees of competency without the need to support these competencies during projects that do not require the same competency profile. Thus, the virtual enterprise has the benefits of a large, well-found enterprise, without having to pay for the maintenance overhead of resources that are not required for other projects.

The concept of agility in a manufacturing context has emerged recently [6, 7]. Most authors agree that "agility" is the ability to respond rapidly to some external and unexpected event. The argument promoting agility is that it enables better survival in turbulent market conditions. Most agile responses, however, are tailored to changes in product demand, either in the form of production levels or in alternate design. The solutions to these tend to fall in line with traditional manufacturing theory (for example, by applying Just-in-Time methods) or design modification (such as mass customisation applied after the initial product launch). Thus, enterprises use the agile methods to enable them to respond to the market, based on a given design.

Where designs require modification, change management methods provide a structure to enable managed modification of a design or manufacture process [8, 7]. There are two aspects of change management: (1) the process of changing a given design, and (2) the design of artefacts such that they are modified more easily when needed. In the process of changing a given design, care must be taken to avoid the possibility that a change in one part of the design might impact negatively on some other part. This requires not only an understanding of how all aspects of the design interact with each other, but also a mechanism to rectify such collisions when they occur. This can be particularly troublesome when the two conflicting design aspects are "owned" by different agents in a virtual enterprise. If change is expected to a design, then the original design can be made such that changes can be applied easily. There is relatively little literature on changing the design mid-way through the design process. This tends to be subsumed under failing a mid-way design check point, thereby sending the design back to an earlier

phase. Such changes, however, arise frequently due to the customer modifying the original specification.

Finally, it is important to consider the human aspects of engineering projects. Where multiple agents are responsible for different but interacting design aspects, it is important that all parties cooperate. Evidence, however, shows that human nature tends to "hide" problems in the hope that they can be resolved without needing to admit there ever was a problem [9]. When the problem is not resolved, it then has an amplified effect on the remainder of the design when it no longer can be hidden. Frequently, resolving these problems requires changes to the design that in turn result in the design process moving backwards. While an agile design process will not change the problems arising due to human factors, it can mitigate the effects of these factors when they arise.

This paper will address how the design process can be made agile. By using concurrent engineering methodology as a starting point, a framework will be proposed that will enable analysis of how unexpected events affect the concurrent design process. Given this analysis tool, it then becomes possible to test different design scenarios to handle an unexpected event occurring during the design process. This framework will be concerned strictly with the temporal aspects of a design project, and not with any other resource use. The underlying principle, however, can be adapted readily to these different resources.

3. Analysis Framework

The basis of this agile design analysis framework stems from concurrent engineering and virtual enterprise methodology. Specifically, given a design project, the project is decomposed into a number of independent tasks that are undertaken by different design agents. The concurrent engineering view assumes that the next global phase of the project then will be the combining of the component solutions into the final deliverable.

The agile design framework considers what happens when some interruption occurs to a design agent. The interruption events to be considered are those that incur a time penalty to the project. Depending on the nature of the event and on how much slack was built into the project, such events potentially result in a global effect to the whole design project. For analysis purposes, the following four levels of (local) event severity have been categorised:

(1) Trivial: The problem can be resolved completely at the local level, and a small time penalty is incurred.

(2) Minor: The problem requires the agent to seek external assistance, or minor redeployment of part of the work to another partner within the virtual enterprise.

(3) Major: The problem cannot be resolved by the agent or by another member of the virtual enterprise. A new member is needed to join the virtual enterprise, and the redeployment of work and initiation of the new member to the project incurs a serious time penalty.

(4) Fatal: The problem cannot be resolved by the agent, and there exists no external agent that can provide support. Effectively, the design is fundamentally flawed and is not realisable.

An event occurring during the design process effectively requires more time to be spent than originally planned. This can be modelled simply by in effect "turning the clock back" on the design process. In effect, this encodes an interruption as being an event that requires redesign work to be performed. So, if an event happens in the detailing stage, the severity of the event determines how far the whole project is set back. For example, a trivial event in the detailing stage might require only the affected agent to restart this stage. The more serious "minor" event might result in the agent returning to the embodiment stage, while a "major" event will result in a new agent starting this part of the design work from the start. Another possibility in the "minor" / "major" cases is that several other design agents are affected, to the respective degrees of the event classification. Finally, in the case of a "fatal" event, it is assumed that the whole design collapses due to the event. In this case, the design requires

fundamental rework. Hence, all agents will start from fresh, effectively under a new virtual enterprise.

3.1 Event Sources

External and unexpected events are the source of interruptions to the design process [10]. This paper shall distinguish three major sources of interrupting events: (1) late customer design brief change, (2) in-production design change, and (3) agent failure. The severity of the event will depend on the nature of the event and the ability of the relevant agent(s) to handle the event.

Customers, in general, do not provide flawless documentation and, depending on circumstances, request unexpected design changes before the end of a project. This customer-driven change requires the design to be reconsidered. At the trivial level, it could be a simple matter of requesting a different material for the external body. More significant requests are those that change the functionality or core design of the artefact.

In-production design changes are those that arise either internally due to identifying additional design constraints or novel design solutions. These can be regarded in a similar manner as customer-driven changes, and have similar characteristics. But because they are internal, there is in general a greater understanding of the nature of the problem.

Finally, agent failure represents the event in which an agent is not capable of delivering the solution as originally specified. Again, the effect of this failure can take the full range between trivial, wherein the receiving agent can absorb the failure and otherwise continue normally through to fatal, wherein the failure of an agent results in the failure of the whole design project. The failure of an agent effectively causes a design change request, which needs to be suitably propagated through the virtual enterprise.

3.2 Concurrent Design Representation

By distributing the design tasks to independent agents, each agent can be seen to perform a classical, single-threaded design project. Using the Pahl and Beitz design phase categories [3], the normal state progression of any agent would be as follows:

Conceptual: determine the nature of the final design,

Embodiment: determine how the conceptual solution takes form,

Detail: determine the exact geometry and other physical aspects of the solution, and

Manufacture: wherein the design is realised.

Depending on the design task and the agent's expertise, each agent will require a different amount of time to complete each stage. The sum of these times represents the total time an agent requires to complete its task, assuming no interruptions. Note that using this scheme does allow for agents to specialise in particular aspects of the design process—for example, a manufacturing agent will require zero time for the prior design activities.

3.3 Agile Design Response

The state of the whole virtual enterprise then is taken as the sum of the agent states. When an interrupting event occurs to an agent, the effect can be propagated through use of the causal links of the global design project within the virtual enterprise. The two extreme event types (trivial and fatal) do not require an agile design methodology to mitigate, as they either are absorbed locally or globally destroy the project. The minor and major event classes, however, do provide an opportunity to apply agile design methods to minimise the total time penalty to the project.

Responding with agility requires that when an event occurs in the concurrent design process, all design agents are able to respond accordingly. In a non-agile system, the penalty is accumulated totally by the affected agent. In the agile system, other agents aim to reduce the impact of the event by being able to self-incur trivial events. Thus, the required change due to the source event is, in effect, mitigated through the relevant partners in

the virtual enterprise. The aim of this analysis framework is to model and simulate such events, so as to be able to test mitigation policies under laboratory conditions.

3.4 Simulation

The agile design framework provides a means for performing simulation. Two assumptions must be made when running such simulations: (1) how far back the event would set back the affected agent, and (2) how the redesign work can be redistributed across the virtual enterprise. It is the second assumption that is the more difficult to make, as this will be dependent on the nature of the event and design project. In order to test redistribution policies, it is sufficient to provide a set of reasonable mappings between event severity and amount of rework that various agents in the virtual enterprise would take on to mitigate the penalty.

In order to test a set of different rework policies (e.g., “don’t mitigate”, “moderate mitigation”, “global mitigation”, etc.), a Monte-Carlo approach is used to run a series of experiments. The result of each experiment is the total time for the project. Policies then can be evaluated by comparing the mean and standard deviation of these total times.

4. Illustration

The simplified scenario of an aeroplane design and manufacturing project can be applied to demonstrate the effects of the scenarios discussed. Consider that the aeroplane design and manufacturing is controlled by Integrator A for Customer 1. The product is split into 5 modules, namely: fuselage, wings, engines, electronic systems, and landing gear. For each module, an agent is brought into the virtual enterprise, so as to design and build the aircraft, and this agent is controlled by Integrator A.

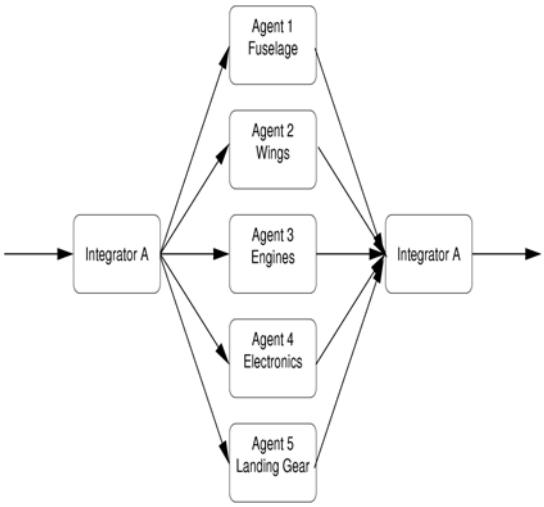

Figure 1: A Simplified Virtual Enterprise for Aircraft Manufacturing.

Each agent within the virtual enterprise follows its own design process, typically following the form illustrated in Figure 1. This scenario now can be applied to illustrate the effects of each of the 4 classifications of external events. It should be considered that the expected time for each agent to complete its work is the same period, i.e., the critical chain is dictated by each agent equally, and a delay to any agent represents a delay to the overall time of the process.

4.1 Trivial Event

Taking agent 1 as the first example, during the product design process, an external event that might impact on the time taken to convert input into output might be a requirements change from the customer. For example, the fuselage is now required to have 4 emergency exits along each side, rather than the original 3 specified.

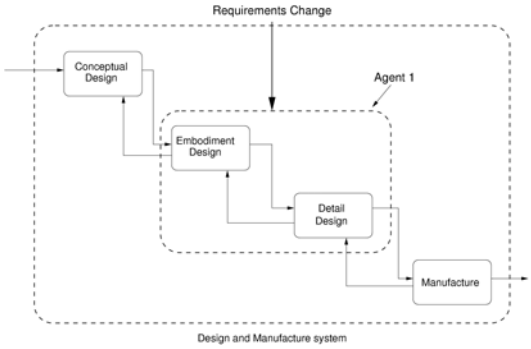

Figure 2: An External Event.

This external event has the effect of reducing how close the agent is to achieving the necessary output because the agent is now further from the desired solution than before the event. Figure 3 illustrates this loss of work.

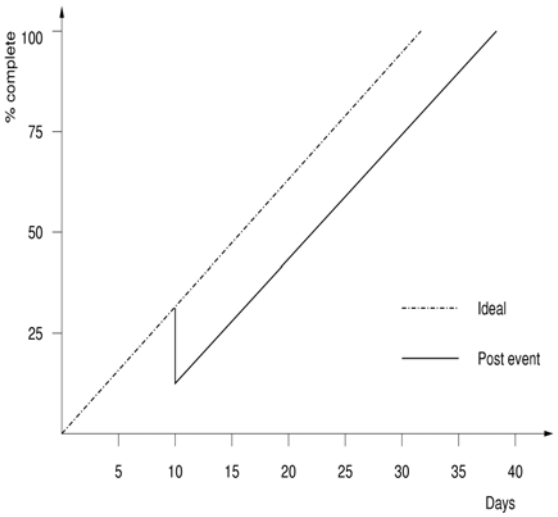

Figure 3: Effect of a Trivial External Event.

Figure 3 illustrates an external event at day 10 of a design process, with a magnitude of 25% of the work already carried out. The time taken for the agent to get back to the position at which it was before the event is, in this case, 8 days. This is defined as T_{Re}, the Time-Response to an external event. The amount of time the process should have taken without the external events is 30 days (T_p).

Therefore, the total time (T_T) is defined as follows:

$$T_T = T_P + \sum_{j=0}^{n_e} T_{Re,j} + \sum_{j=0}^{n_i} T_{Ri,j} \qquad (1)$$

wherein T_{Re} is the time-response for each external event and T_{Ri} is the time-response for each internal event (internal feedback), n_e and n_i are the number of external and internal events, respectively: T_T=38 days in this example.

The event illustrated is classified as a trivial event because it can be dealt with in-house, with a minimal Time-Penalty. This is not always the case, however. Empirical evidence suggests that events often require the use of additional resources, often of an expert nature, to resolve problems raised by unpredictable external events.

4.2 Minor Event

Minor events describe events dictating that the agents no longer have all the skills, resources or knowledge to develop an in-house solution, or that to do so would be more costly than the use of expert help. In this scenario, the agent must identify, rapidly, a partner with the necessary skills, resources or knowledge to “plug the gap” created by the event. This is where the agility of the process will allow the response time to be reduced and the benefits exploited.

Agent 2 is responsible for the design and manufacture of the wings for the new aircraft. An external event, however, dictates that the wing panels no longer should be riveted, but instead welded to save weight, due to new technology becoming available. In this instance, Agent 2 does not possess the skills and knowledge to weld the wing panels satisfactorily, but the customer demands it for weight and, thereby, fuel and cost savings. Agent 2 still has the core competences to design and manufacture the rest of the wing structure, and so expert help is sought to assist specifically with the welding. This assistance normally would be found within the virtual enterprise, but could be assistance brought in from outside the enterprise if the work were to be assistance rather than taking control of all or part of a process.

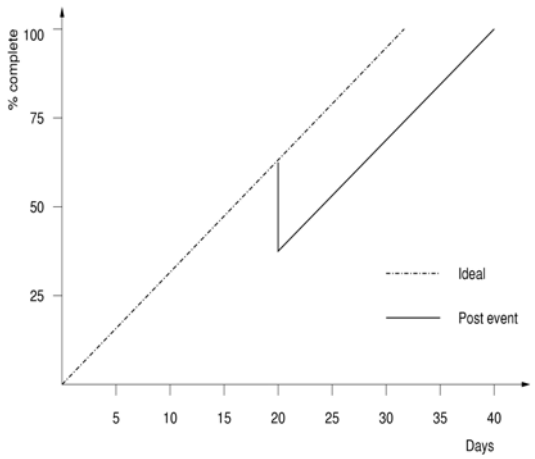

Figure 4: Effect of a Minor External Event.

This minor event has a more significant "knock-back" effect on the work done, and therefore imposes a greater Time-Penalty on the process.

For the minor event above, the total time is 41 days instead of the perfect 30 days. If an agile design process can reduce the effort lost from 35% of work to 15%, the Time-Penalty can be reduced to 5 days, a savings of 6 days. Figure 5 shows this effect.

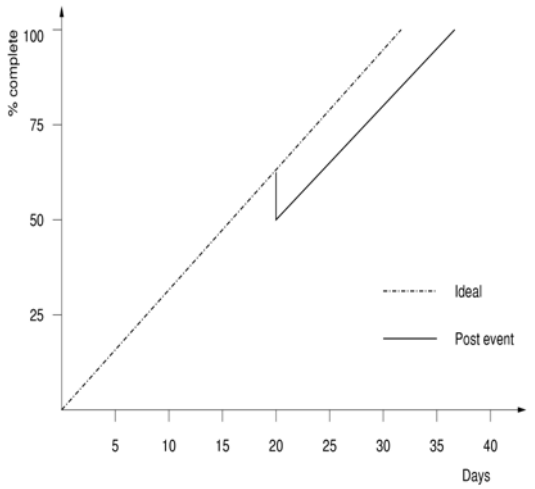

Figure 5: An Agile Design Time-Response to a Minor External Event.

4.3 Major Event

A major event is defined by the failure of an agent to satisfy the demands put upon it by the greater organisation or system; it cannot convert its inputs into the required outputs.

An example scenario similar to one seen in reality might be Agent 3, responsible for the engines, failing to design and manufacture an engine sufficiently powerful to satisfy the customer requirements it was given. The Agent simply does not have the knowledge, experience and / or resources to meet the demands. This is classified as a major external event, which to the rest of the virtual enterprise is unpredictable. The Time-Penalty is likely to be serious, as the work completed is likely to be reset to zero. An alternative scenario might be that an agent goes bankrupt during the project and no longer can continue its role.

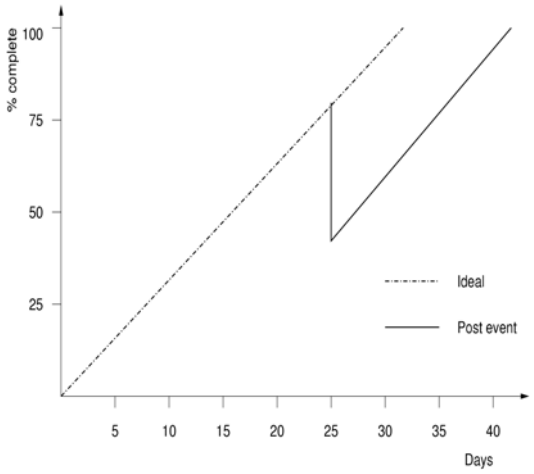

Figure 6: Time Response to a Major Event.

In this situation, the agent is deemed to have failed and another agent is required to take control of that specific process and to join the virtual enterprise. The speed with which this can be achieved is a function of the agility of the organisation as a whole. The example shown below shows a 24-day delay to a process that should have taken 30 days, due to unpredictable failure after 24 days.

4.4 Fatal Event

The fatal event is an external event that is catastrophic to the design as a whole, not just to any individual agent. An example of a fatal event might be the introduction of legislation stating

that emissions of greenhouse gases for any mode of transport must be below a given level. This event would impact directly the agent responsible for the engines—in our case example, Agent 3; the effect, however, is that the design of the overall process is fundamentally flawed. No replacement of Agent 3 will be able to produce the output required for the project to be realised.

These 4 classifications of event cover a broad spectrum of real-life events, for which agile design will be of significant benefit, by reducing to a minimum the time-response of an agent or overall system, and thus the time-delay of whole projects.

5. Discussion

The agile design methodology does assume, and requires, that the design project is being undertaken in a virtual enterprise environment. The aim of the agile design methodology is to enable rapid redeployment of design work when required due to an unforseen event occurring. These events are basically design change requirements, arising due either to customer request, internal request resulting from design work, or internal request due to an agent failing to fulfil its task. A classification for the severity of these events was proposed that covered the full spectrum. Within this severity spectrum, it has been argued that the extreme classes are not appropriate for agile design methods. Events classified away from these extremes, however, provide an opportunity for agile response.

The agile design methodology was illustrated using a simplified aircraft design scenario. While the design process in this case was completely parallel, it demonstrated how the design work could be redistributed in each of the event classes. In the agile cases, it was argued that this provided a more rapid resolution to the events than would have been possible otherwise.

6. Conclusions

This paper introduced a framework for representing and analysing agile design methodology based on concurrent engineering. This illustrated how the design state could be mapped out onto the virtual enterprise network, and provided a means for representing and measuring the effect of how unexpected design events occur during the design process. The agile design methodology redistributes the design work after an event occurs according to the severity. The proposed framework enables virtual enterprises to test various work redistribution policies, using Monte Carlo methods to determine the most suitable for different event types.

The next phase of this research will verify the event classifications with SMEs that have experience operating in virtual enterprises. This will include reviewing case studies wherein agents have encountered serious unplanned redesign work during the design process, and reviewing how this rework would have been distributed across the virtual enterprise.

One aspect that has not been considered is the communication latency within a virtual enterprise. It has been assumed that when work is redistributed between agents, this results only in the affected agents “turning back the clock” on their design process. It was assumed that this resetting of the design process incorporated any latency rather than explicitly accounting for this.

References

1. Kusar, J., Duhovnik, J., Grum, J. and M. Starbek, 2004, "How to Reduce New Product Development Time," Robotics and Computer-Integrated Manufacturing, **20**, 1-15.

2. Camarinha-Matos, L. M. and H. Afsarmanesh, 2003, "Elements of VE Infrastructure," Computers in Industry, **51**, 139-163.

3. Pahl, G. and W. Beitz, 1996, *Engineering Design: A Systematic Approach*, Springer-Verlag London, second edition.

4. Carter, D. E. and B. S. Baker, 1991, *Concurrent Engineering: The Product Development Environment for the 1990s*, Addison-Wesley.

5. Armoutis, N. D. and J. Bal, 2003, *Building the Knowledge Economy: Issues, Applications, and Case Studies*, chapter E-Business through Competence Profiling, 474–482. IOS Press.

6. Lau, H. C. W., Wong, C. W. Y., Pun, K. F. and K. S. Chin, 2003, "Virtual Agent Modelling of an Agile Supply Chain Infrastructure, Management Decision, **41**(7), 625-634.

7. Jiang, Z. and R. Y. K. Fung, 2003, "An Adaptive Agile Manufacturing Control Infrastructure Based on TOPNs-CS Modelling," International Journal of Advanced Manufacturing Technology, **22**, 191-215.

8. Terwiesch, C., Loch, C. H. and A. De Meyer, 2002, "Exchanging Preliminary Information in Concurrent Engineering: Alternative Coordination Strategies," Organization Science, **13**(4), 402-419.

9. Ford, D. N. and J. D. Sterman, 2003, "The Liar's Club: Concealing Rework in Concurrent Development," Concurrent Engineering: Research and Applications, **11** (3), 211-219.

10. Ye, N., 2002, "Information Infrastructure of Engineering Collaboration in a Distributed Virtual Enterprise," International Journal of Computer Integrated Manufacturing, **15** (3), 265-273.

Biographies

Peter Matthews received his B.A. in Mathematics and a Ph.D. in the application of A.I. tools at the University of Cambridge. He is now the Lecturer in Design Informatics at the School of Engineering at the University of Durham, researching the application of stochastic and other modelling techniques to the early engineering design phases.

Chris Lomas graduated from the University of Durham in 2003 with a Masters degree in Engineering. He is a Ph.D. candidate and his research interest relates to Agile Design.

Nikolaos Armoutis is a Research Fellow in the School of Engineering at Durham University. He is currently involved in identifying and assessing competencies in engineering companies with a view to forming collaborating alliances and virtual organizations.

Paul Maropoulos is a Professor of Engineering at the Innovative Manufacturing Research Centre at Bath University. He is director of the Global Digital Enterprise Research Laboratory (GDERL)